Amendments to the Claims

This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims

1. (Currently amended) A computer, comprising:

a processor pipeline designed to alternately execute instructions coded for first and second different computer architectures or coded to implement first and second different processing conventions;

a memory for storing instructions for execution by the processor pipeline, the memory being divided into pages for management by a virtual memory manager, a single address space of the memory having first and second pages;

a memory unit designed to fetch instructions from the memory for execution by the pipeline, and to fetch stored indicator elements associated with respective memory pages of the single address space from which the instructions are to be fetched, each indicator element designed to store an indication of which of two different computer architectures ~~and/or execution~~ or processing conventions under which instruction data of the associated page are to be executed by the processor pipeline, the first architecture having a pre-defined, established definition, the computer providing ~~a faithful~~ an implementation of the first architecture;

the memory unit ~~and/or~~ or processor pipeline further designed to recognize an execution flow from the first page, whose associated indicator element indicates the

first architecture or ~~execution~~ the first processing convention, to the second page, whose associated indicator element indicates the ~~first~~ second architecture or ~~execution~~ the second processing convention, and in response to the recognizing, to adapt a processing mode of the processor pipeline or a storage content of the memory to effect execution of instructions in the architecture ~~and/or~~ or under the processing convention indicated by the indicator element corresponding to the instruction's page.

2. (Currently amended) The computer of claim 1, wherein:

~~wherein~~ the two computer architectures are two instruction set architectures; and

~~and wherein~~ the adapting includes controlling instruction execution hardware of the computer to interpret the instructions according to the two instruction set architectures according to the indicator elements.

3. (Currently amended) The computer of claim 1, wherein the two processing conventions are first and second calling conventions, and further comprising:

hardware ~~and/or~~ or software designed to recognize when program execution has flowed or transferred from a region whose indicator element indicates the first calling convention to a region whose indicator element indicates the second calling convention, and in response to the recognition, to alter the data storage content of

the computer to create a program context under the second calling convention that is logically equivalent to a pre-alteration program context under the first calling convention.

4. (Currently amended) A method, comprising ~~the steps of~~:

executing instructions fetched from first and second regions of a memory of a computer, the instructions of the first and second regions being coded for execution by computers of first and second architectures or following first and second data storage conventions, respectively, the memory regions having associated first and second indicator elements, the indicator elements each having a value indicating the architecture or data storage convention under which instructions from the associated region are to be executed, the first architecture having a pre-defined, established definition, the computer providing ~~a faithful~~ an implementation of the first architecture;

when execution of the instruction data flows or transfers from the first region to the second region, adapting the computer for execution in the second architecture or data storage convention.

5. (Original) The method of claim 4, wherein:

the regions are pages managed by a virtual memory manager.

6. (Original) The method of claim 5, wherein the indicator elements are stored in a table of indicator elements distinct from a primary address translation table used by the virtual memory manager, the indicator elements of the table associated with corresponding pages of the memory.

7. (Original) The method of claim 5, wherein the indicator elements are stored in a table, each indicator element associated with a corresponding physical page frame.

8. (Canceled)

9. (Original) The method of claim 4, wherein the regions are lines of an instruction cache.

10. (Currently amended) The method of claim 4, wherein:

~~wherein~~ the two architectures are two instruction set architectures; and

~~and wherein~~ the adapting ~~step~~ includes controlling instruction execution hardware of the computer to interpret the instructions according to the two instruction set architectures according to the indicator elements.

11. (Original) The method of claim 10, wherein:

the regions are pages managed by a virtual memory manager.

12. (Previously presented) The method of claim 11, wherein the indicator elements are stored in a table whose entries are indexed by physical page frame number.

13. (Canceled)

14. (Currently amended) The method of claim 10, wherein a mode of execution of the instructions is changed without software intervention when execution flows or transfers from the first region to the second region.

15. (Currently amended) The method of claim 10, wherein execution of the computer takes an exception when execution flows or transfers from the first region to the second region.

16. (Currently amended) The method of claim 15, wherein [[the]] a mode of execution of the instructions is explicitly controlled by an exception handler.

17. (Currently amended) The method of claim 10, wherein:

one of the regions stores an ~~off-the-shelf~~ operating system binary coded in an instruction set non-native to the computer, the non-native instruction set providing access to a reduced subset of the resources of the computer.

18. (Currently amended) The method of claim 10, ~~wherein the two conventions are first and second data storage conventions, and~~ further comprising ~~the step of~~:

recognizing when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indicator element indicates the second data storage convention, and in response to the recognition, altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention.

19. (Currently amended) The method of claim 18, wherein:

the regions are pages managed by a virtual memory manager; and

one of the two data storage conventions is a register-based calling convention, and the other data storage convention is a memory stack-based calling convention.

20. (Currently amended) The method of claim 18, further comprising ~~the steps of~~:

classifying control-flow instructions of a computer instruction set into a plurality of classes; and

during execution of a program on [[a]] the computer, as part of the execution of instructions of the instruction set, updating a record of the class of the classified control-flow instruction most recently executed;

the adjusting process being determined, at least in part, by the instruction class record.

21. (Currently amended) The method of claim 18, ~~wherein:~~

wherein the instruction data coded for execution by a first of the two instruction set architectures observes a first data storage convention associated with the first architecture, and instruction data coded for execution by a second of the two instruction set architectures observes a second~~, different,~~ data storage convention associated with the second architecture, the second data storage convention being different from the first data storage convention, a single indicator element indicating both the instruction set architecture and the data storage convention; and

~~and~~ further comprising ~~the step of~~ recognizing when program execution flows or transfers from a region using the first instruction set architecture to a region using the second instruction set architecture, and in response to the recognition, adjusting the data storage content of the computer from the first data storage convention to the second data storage convention.

22. – 33. (Canceled)

34. (Currently amended) ~~The~~ A method ~~of claim 22, further~~ comprising ~~the step of~~:

executing instructions fetched from first and second regions of a memory of a computer, the instructions of the first and second regions being coded for execution by computers following first and second data storage conventions, the memory regions having associated first and second indicator elements, the indicator elements each having a value indicating the data storage convention under which instructions from the associated region are to be executed;

recognizing when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indictor element indicates the second data storage convention, and in response to the recognition, altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention;

classifying control-flow instructions of a computer instruction set into a plurality of classes; and

during execution of a program on [[a]] the computer, as part of the execution of instructions of the instruction set, updating a record of the class of the classified control-flow instruction most recently executed;

the ~~adjusting~~ altering process being determined, at least in part, by the instruction class record.

35. (Original) The method of claim 34, wherein:

one of the two data storage conventions is a register-based calling convention, and the other data storage convention is a memory stack-based calling convention.

36. (Currently amended) The method of claim 34, wherein:

in some of the control-flow instructions, the classification is statically determined by [[the]] an opcode of the instructions; and

in other of the control-flow instructions, the classification is dynamically determined based m a full/empty status of a register.

37. (Currently amended) A computer processor, comprising:

a processor pipeline configured to alternately execute instructions ~~of computers~~ of two different architectures or processing conventions; and

a memory unit designed to fetch instructions from a computer memory for execution by the pipeline, and to fetch stored indicator elements associated with respective memory regions of a single address space from which the instructions are to be fetched, each indicator element designed to store an indication of the architecture or ~~execution~~ processing convention under which the instruction data of the associated region are to be executed by the ~~processor~~ pipeline, the first architecture having a predefined, established definition, the computer providing ~~a faithful~~ an implementation of the first architecture;

the memory unit ~~and/or~~ or processor pipeline further designed to recognize an execution flow or transfer from a region whose indicator element indicates one architecture or ~~execution~~ processing convention to another.

38. (Currently amended) The computer processor of claim 37, wherein the indicator elements are stored in a table of indicator elements distinct from a primary address translation table used by [[the]] a virtual memory manager, the indicator elements of the table associated with corresponding pages of the memory.

39. (Currently amended) The computer processor of claim 37, further comprising:

a translation look-aside buffer (TLB); and

TLB control circuitry designed to load the indicator elements into the TLB from a table stored in memory, the entries of the table being indexed by ~~associated with~~ corresponding physical page frame number.

40. (Currently amended) The computer processor of claim 37,

wherein the two architectures are two instruction set architectures, and

further comprising[[:]] processor pipeline control circuitry designed to control the ~~processor~~ pipeline to effect interpretation of the instructions under the two instruction set architectures alternately, according to the associated indicator elements.

41. (Currently amended) The computer processor of claim 40, further comprising:

software programmed to manage a transition between the execution of a program executing in the first instruction set architecture, being an instruction set architecture native to the computer processor, and execution of an ~~off-the-shelf~~ operating system coded in the second instruction set, being an instruction set non-native to the computer[[,]] and providing access to a reduced subset of the resources of the computer.

42. (Currently amended) The computer processor of claim [[40]] 41:

wherein each indicator element ~~being~~ is further designed to store an indication of a calling convention under which the instruction data of the associated region [[are]] is coded for execution by the ~~processor~~ pipeline; and

[[and]] further comprising software programmed to alter the data storage content of a computer using the computer processor to create a program context under the second calling convention that is logically equivalent to a pre-alteration program context under the first calling convention;

the memory unit further designed to recognize when program execution has flowed or transferred from a region whose indicator element indicates the first calling convention to a region whose indicator element indicates the second calling convention, and in response to the recognition, to invoke the transition management software.

43. (Currently amended) The computer processor of claim 42, wherein the memory unit is <u>further</u> designed to recognize a single indicator element to indicate both the instruction set architecture and calling convention of a region.

44. (Original) The computer processor of claim 42, wherein the memory unit and software are designed to effect a transition between instruction boundaries, between execution in a region coded in the first instruction set using the first calling convention to execution in a region coded in the second instruction set using the second calling convention, so that code at the source of the flow or transfer may effect the execution transition without being specially coded for code at the destination.

45. (Original) The computer processor of claim 42, wherein:

one of the two calling conventions is a register-based calling convention, and the other calling convention is a memory stack-based calling convention.

46. (Original) The computer processor of claim 42, wherein the logical resources for support of the first and second instruction set architectures are overlaid on the physical resources of the computer processor according to a mapping that assigns corresponding resources of the two architectures to a common physical resource when the resources serve analogous functions in the calling conventions of the two architectures.

47. (Currently amended) The computer processor of claim 42, wherein

a rule for altering the data storage content from the first calling convention to the second calling convention is determined by examining a descriptor associated with the location of execution before the recognized execution flow or transfer.

48. (Currently amended) The computer processor of claim 42, wherein control-flow instructions of the instruction set are classified into a plurality of classes; and

the ~~processor~~ pipeline updates a record of the class of the classified control-flow instruction most recently executed;

the storage alteration process being determined, at least in part, by the instruction class record.

49. (Original) The computer processor of claim 40, further comprising:

a transition manager designed to effect a transition between the execution of code coded in instructions of a first instruction set architecture and code coded in instructions of a second instruction set architecture, the transition manager designed to alter a bit representation of a datum from a first representation under the first architecture to a second representation under the second architecture, the alteration of representation being chosen to preserve the meaning of the datum across the change in execution architecture.

50. (Currently amended) The computer processor of claim 40, further comprising:

circuitry designed to raise an exception when the computer recognizes that execution has flowed or transferred from a region whose indicator element indicates one architecture or execution convention to another.

51. – 59. (Canceled)

60. (Currently amended) [[The]] A method ~~of claim 54, further~~ comprising:

storing instructions in pages of a computer memory managed by a virtual memory manager, the instruction data of the pages being coded for execution by, respectively, computers of two different architectures or under first and second data storage conventions;

in association with pages of the memory, storing corresponding indicator elements indicating the architecture or convention in which the instructions of the pages are to be executed, the pages' indicator elements being stored in a table whose entries are indexed by physical page frame number;

executing instructions from the pages in a common processor, the processor designed, responsive to the page indicator elements, to execute instructions in the architecture or under the convention indicated by the indicator element corresponding to the instruction's page;

recognizing when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indicator element indicates the second data storage convention, and in response to the recognition, altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention;

classifying control-flow instructions of a computer instruction set into a plurality of classes; and

during execution of a program on [[a]] the computer, as part of the execution of instructions of the instruction set, updating a record of the class of the classified control-flow instruction most recently executed;

the altering process being determined, at least in part, by the instruction class record.

61. (Canceled)

62. (Canceled)

63. (Currently amended) A microprocessor chip, comprising:

an instruction unit, configured to fetch instructions from a memory managed by the virtual memory manager, and configured to execute instructions coded for

first and second different computer architectures or coded to implement first and second different data storage conventions;

the microprocessor chip ~~being~~ designed ~~(a)~~ to:

retrieve indicator elements stored in association with respective pages of the memory, each indicator element indicating the architecture or convention in which the instructions of the page are to be executed, ~~and (b) to~~

recognize when instruction execution has flowed or transferred from a page of the first architecture or convention to a page of the second architecture or convention, as indicted by the respective associated indicator elements, and

~~(c) to~~ alter a processing mode of the instruction unit or a storage content of the memory to effect execution of instructions in accord with the indicator element associated with the page of the second architecture or convention;

wherein the indicator elements are stored in ~~a table~~ an instruction cache distinct from a primary address translation table used by a virtual memory manager, the indicator elements of the table being stored in association with respective pages of the memory.

64. – 68. (Canceled)

69. (Currently amended) The microprocessor chip of claim 63, wherein the microprocessor chip ~~being~~ is designed to raise an exception when execution flows or transfers from the first region to the second region; and

[[and]] further comprising an exception handler ~~software programmed~~ designed to handle the exception by explicitly controlling a mode of execution of the instructions.

70. (Canceled)

71. (Currently amended) The microprocessor chip of claim 63, wherein:

the two architectures are two instruction set architectures, and

the microprocessor chip controls the instruction unit to interpret the instructions according to the two instruction set architectures according to the indicator element corresponding to the pages from which the instructions are fetched.

72. (Currently amended) The microprocessor chip of claim 71, further comprising:

software programmed to manage a transition between the execution of a program executing in the first instruction set architecture, being an instruction set architecture native to the computer processor, and execution of an ~~off-the-shelf~~ operating system coded in the second instruction set, being an instruction set non-native to the computer and providing access to a reduced subset of the resources of the computer.

73. (Currently amended) The microprocessor chip of claim 71:

wherein each indicator element ~~being~~ is further designed to store an indication of a data storage convention under which the instruction data of the associated page [[are]] is coded for execution by the instruction unit; and

[[and]] further comprising software programmed to alter the data storage content of a computer using the microprocessor chip, to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention;

the microprocessor chip further designed to recognize when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indicator element indicates the second data storage convention, and in response to the recognition, to invoke the transition management software.

74. (Canceled)

75. (Original) The microprocessor chip of claim 73, further designed to recognize a single indicator element to indicate both the instruction set architecture and calling convention of a page.

76. – 85. (Canceled)

86. (Currently amended) The microprocessor chip of claim 63, wherein control-flow instructions of the microprocessor's instruction set are classified into a plurality of classes; and

the ~~fetch-and-execute~~ instruction unit updates a record of the class of the classified control-flow instruction most recently executed;

the storage alteration process being determined, at least in part, by the instruction class record.

87. – 112. (Canceled)

113. (Currently amended) A computer processor, comprising:

a processor pipeline configured to alternately execute instructions ~~of computers~~ of two different architectures; and

a memory unit designed to fetch instructions from a computer memory for execution by the pipeline, and to fetch stored indicator elements associated with respective ~~necessarily-disjoint~~ memory regions of a single address space from which the instructions are to be fetched, each indicator element designed to store an indication of the architecture under which the instruction data of the associated region are to be executed by the ~~processor~~ pipeline, and to store a separate indicator element designed to indicate a data storage convention observed by instructions of the associated region;

the memory unit ~~and/or~~ or ~~processor~~ pipeline further designed to recognize an execution flow or transfer from a region whose indicator element indicates one instruction set architecture to another; and

hardware ~~and/or~~ or software designed to recognize when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indicator element indicates the second data storage convention, and in response to the recognition, to alter the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention.

114. (Currently amended) The computer processor of claim 113, wherein the two architectures are two instruction set architectures, and further comprising:

processor pipeline control circuitry designed to control the ~~processor~~ pipeline to effect interpretation of the instructions under the two instruction set architectures alternately, according to the associated indicator elements.

115. (Currently amended) The computer processor of claim 114, further comprising:

software programmed to manage a transition between the execution of a program executing in the first instruction set architecture, being an instruction set

architecture native to the computer processor, and execution of an ~~off-the-shelf~~ operating system coded in the second instruction set, being an instruction set non-native to the computer[[,]] and providing access to a reduced subset of the resources of the computer.

116. (Currently amended) The computer processor of claim [[114]] 115:

wherein the data storage convention indicator elements ~~being~~ are designed to store an indication of a calling convention under which the instruction data of the associated region are coded for execution by the ~~processor~~ pipeline; and

[[and]] further comprising software programmed to alter the data storage content of a computer using the computer processor to create a program context under [[the]] a second calling convention that is logically equivalent to a pre-alteration program context under [[the]] a first calling convention;

the memory unit is further designed to recognize when program execution has flowed or transferred from a region whose indicator element indicates the first calling convention to a region whose indicator element indicates the second calling convention, and in response to the recognition, to invoke the transition management software.

117. (Canceled)

118. (Canceled)

119. (Currently amended) The computer processor of claim [[120]] 116, wherein the memory unit and software are designed to effect a transition between instruction boundaries, between execution in a region coded in the first instruction set using the first calling convention to execution in a region coded in the second instruction set using the second calling convention, so that code at the source of the flow or transfer may effect the execution transition without being specially coded for code at the destination.

120. (Canceled)

121. (Currently amended) The computer processor of claim [[120]] 116, wherein:

one of the two calling conventions is a register-based calling convention, and the other calling convention is a memory stack-based calling convention.

122. (Currently amended) The computer processor of claim [[120]] 116, wherein the logical resources for support of the first and second instruction set architectures are overlaid on the physical resources of the computer processor according to a mapping that assigns corresponding resources of the two architectures to a common physical resource when the resources serve analogous functions in the calling conventions of the two architectures.

123. (Currently amended) The computer processor of claim [[120]] 116, further comprising:

a transition manager designed to effect a transition between execution under the first calling convention to execution under the second calling convention, the transition manager designed to alter a bit representation of a datum from a first representation to a second representation, the alteration of representation being chosen to preserve the meaning of the datum across the change in calling convention.

124. (Currently amended) The computer processor of claim [[120]] 116, further comprising:

software ~~and/or~~ or hardware designed to copy a datum from a first location to a second location, the first location having a use under the first calling convention analogous to the use of the second location under the second calling convention.

125. (Currently amended) The computer processor of claim 124, further comprising:

software ~~and/or~~ or hardware designed to copy a datum from a third location to a fourth location, the third location having a use under the first calling convention analogous to the use of the third location under the first calling convention and to the fourth location under the second calling convention, the software ~~and/or~~ or hardware for the copying being programmed to assume that

exactly one of the first and third locations is no longer required by the execution of the program.

126. (Currently amended) The computer processor of claim [[120]] 116, wherein

a rule for altering the data storage content from the first calling convention to the second calling convention is determined by examining a descriptor associated with the instruction before the recognized execution flow or transfer.

127. (Currently amended) The computer processor of claim [[120]] 116, wherein control-flow instructions of the instruction set are classified into a plurality of classes; and

the ~~processor~~ pipeline updates a record of the class of the classified control-flow instruction most recently executed;

the storage alteration process being determined, at least in part, by the instruction class record.

128. (Previously presented) The computer processor of claim 113, wherein:

the regions are pages managed by a virtual memory manager.

129. (Previously presented) The computer processor of claim 113, wherein the indicator elements are stored in virtual address translation table entries.

130. (Currently amended) The computer processor of claim 113, wherein the indicator elements are stored in a table of indicator elements distinct from a primary address translation table used by [[the]] a virtual memory manager, the indicator elements of the table associated with corresponding pages of the memory.

131. (Previously presented) The computer processor of claim 113, wherein the indicator elements are stored in respective entries of a table whose entries are indexed by physical page frame number.

132. (Currently amended) The computer processor of claim 113, wherein:

the indicator elements are stored in storage that is architecturally addressable when the ~~processor~~ pipeline is executing in one of the ~~computer~~ architectures ~~or processing conventions,~~ and architecturally unaddressable when the ~~processor~~ pipeline is executing in the other architecture ~~or convention~~.

133. (Currently amended) The computer processor of claim 113, further comprising:

circuitry designed to raise an exception when execution flows or transfers from a region whose indicator element indicates one architecture ~~or execution convention~~ to another.